ATARI.RSC
The Atari Developer's Resource

Vol. VI, Issue 1
January-February 1992

NAMM Report

Mike Fulton

The Anaheim Convention Center was the site of the recent NAMM (National Association of Music Merchants) convention. For three days, January 17-19, music dealers from all over the world examined the latest in musical instrument and recording technology. Throughout all three days, Atari was host to one of the busiest booths on the entire show floor.

Atari was the only major computer manufacturer at the show. Apple and Commodore were not present at all. And although IBM had a small booth supposedly featuring "Multimedia", in reality they were mostly showing games that happened to feature MIDI output and/or sound card support, which didn't appear to hold much interest for the NAMM crowd.

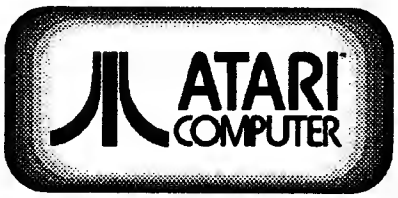
Almost half of the Atari booth was devoted to a special sound room with hourly demonstrations from developers such as Fostex, Hybrid Arts, Steinberg-Jones, C-Lab, and Hotz Technologies.

Included in the demonstrations were excellent performances and talks with musicians such as Mick Fleetwood of Fleetwood Mac, Genesis tour drummer Chester Thompson, composer Terry Riley, and others.

Other music celebrities such as guitar virtuoso Lee Ritenour, movie sound effects wizard Scott Gershin, Mike Pinder from the Moody Blues, the Arsenio Hall Show's Starr Parodi & Jeff Fair, and bassist Alex Acuna also stopped by to see what the excitement was all about.

Nevin Shalit, of Step Ahead Software and current president of the Independent Association of Atari Developers, also gave a seminar each day about non-music software for musicians, centered around productivity and desktop publishing.

The other half of the Atari booth was devoted to a number of different demonstration stations. In addition to stations manned by the developers who were also featured



in the Sound Room, other stations were staffed by developers such as Codehead Software showing their MIDI Spy package, Roland Corp showing their latest music hardware. Also present were software distributors Pacific Software Supply and Thinkware, showing some of the products they distribute, both music related and non-music related.

Thinkware is one of the leading distributors of computer software to music merchants. They also carry a selection of non-music related software. Keep in mind that a good percentage of Atari's dealers are music stores, not computer stores. If you have a product that you'd like to get into the music stores that carry the Atari computer line, then you should contact Ray Bachand at Thinkware, at (415) 255-2091.

NAMM is not just an exposition for manufacturers to show off their products. It's a show to write sales orders. And sales orders and new dealers were being written up at a record pace in the Atari booth.

There were also several STBooks on hand at the show. Although the new notebook machine wasn't being formally displayed at the show, it was shown off by Bill Rehbock at the initial show-starting press conference, as well as shown informally to anybody who asked about it.

A special music issue of the Atari Explorer magazine was previewed at NAMM, featuring The Arsenio Hall Show's Starr Parodi and Jeff Fair on the front cover. Inside the issue is a special 40-page Atari ARTIST insert, with music-related news, interviews, and information.

The syndicated television show "Computer Chronicles" spent a good part of Saturday in the Atari booth, taping the demonstrations in the sound room and conducting interviews throughout the booth.

The footage will be featured in a special episode centered on the use of computers in the music industry. The show will be aired the week of February 25. Please check your local listings for stations and airdates.

Inside This Issue

ATARI.RSC
The Resource File

TECHNICAL DIRECTOR
Bill Rehbock (408) 745-2082

DEVELOPER TECHNICAL SUPPORT
J. Patton (408) 745-2135
GEnie: ATARIDEV
CIS: 70007,1072

Mike Fulton (408) 745-8821
GEnie: MIKE-FULTON
CIS: 75300,1141

Fax: (408) 745-2094

DEVELOPER ADMINISTRATOR
Gail Bacani (408) 745-2022

PORTFOLIO MARKETING MANAGER
Don Thomas (408) 745-2031

SOFTSOURCE ADMINISTRATOR
Dan McNamee (408) 745-2024

CONFIDENTIALITY

The information in ATARI.RSC is confidential and provided for your use in developing products compatible with Atari computers only. You are responsible for protecting the confidentiality of this material, in keeping with your Confidentiality Agreement. If you need to reveal some of the information in this newsletter, contact Bill Rehbock first to get permission.

Copyright © 1992 Atari Corp.
All rights reserved.

Atari Computer Corp.
1196 Borregas Ave.
Sunnyvale, CA 94089-1302

Atari, the Atari logo, TT, and MEGA are trademarks of Atari Corporation.

ATARI.RSC edited by Mike Fulton

ATARI.RSC was created using Pagestream v2.1 on the Atari TT030 computer & Atari TTM194 monochrome monitor, and was printed using Ultrascript on the Atari SLM804 Laser Printer. Some fonts provided courtesy of Computer Safari.

Developer Report

Gail Bacani

Dig that newsletter mailing envelope out of the trash can and have a look at your mailing label. Do you see a five or six digit number after the contact name? This is your personal Developer Identification Number (or Dev. ID).

ST/TT developers' numbers begin with "1" and are five digits in length; Portfolio developers' numbers begin with "P" and are six digits in length. If your number appears shorter, the label probably wasn't long enough to print both contact and number fields. Please contact me for the remainder of your number.

Your Dev. ID is helpful to know for accessing online developer roundtables, when sending in sales orders, or just to carry around in your wallet with your other numbers.

Special Note:

Thank you to all the developers who called in congratulatory messages on my recent marriage. Gail Johnson has been upgraded to Gail Bacani.

Documentation Correction

There is an error in the GEM AES Documentation for the wind_get() function. On page 11-20 there is a reference for the WF_SCREEN value used for the wi_gfield parameter which specifies that:

```
wi_gw1 - low word of address
wi_gw2 - high word of address
wi_gw3 - low word of length
wi_gw4 - high word of length
```

This is not correct. The correct information is:

```
wi_gw1 - high word of address
wi_gw2 - low word of address
wi_gw3 - high word of length
wi_gw4 - low word of length
```

Atari Softsource

Dan McNamee

As you may (or may not) already know, Atari Softsource is alive and well on GEnie. User response to it so far seems to be pretty good, and we are currently averaging about 200 user accesses per month. But developer response in the from of product listings has not been, shall we say, overwhelming. Softsource itself (both the database and unarchived versions of the demos) takes up close to 45 megs on my hard drive, but we are now going to push ahead.

I am now preparing Softsource for mastering on CD. There is still time to get in on the first CD, however. Please, do your entries now.

On the subject of entries, I have written a Superbase Professional application for creating Softsource entries. I will make this available to anyone who wants it. However you will need to provide your own copy of Superbase Professional, or the Superbase runtime package since we do not as of yet have rights to distribute the runtime. But we are working on it. Anyone who is interested in receiving this, please drop me an email on GEnie at D.MCNAMEE, Compuserve at 70007,5166, or call me at (408) 745-2024 during normal business hours.

Please, don't miss out on being included on the first CD. Don't assume that someone else has or will do your entries for you. If you didn't do entries yourself for your products, check yourself to see that they have been done.

Portfolio News

J. Patton & Don Thomas

Portfolio Bug

While testing software, we came across a situation which you should all be aware of.

We issued DOS Fn 25h calls (Absolute Read - to determine there was a RAM card available) and DOS Fn 26h (Absolute Write - to make sure it wasn't write protected) so we could handle DOS critical errors involving the card ourselves. Since the Portfolio does not have hardware to detect if cards are changed, a RAM card could be changed in drive A and subsequent writing to that card would write the older card's FAT. You should issue a DOS Fn 0Dh (Flush Buffers) to force the system to re-access the card to update the FAT.

Optimizing RAM Card Storage

1. PK-lite has been reported to give good results on the Portfolio. PK-lite is a shareware product available for download which will compress your executable code. The program will uncompress when it is loaded into TPA.

2. Format the card on a PC card drive forcing the cluster size to be smaller by using the cluster switches. For a 128k card with many files this can turn out to be a significant amount of space since the default cluster size is 512 bytes and you can change that to 128 bytes.

Extension ROMs

If you have a peripheral ypu can take advantage of extension ROMs to keep your software on without using a card. Extension ROMs can as part of their startup install a launcher onto drive C: whenever the machine is reset and the peripheral is attached. The job of this launcher would be to swap in the ROM space through Int 61 Fn 25.

If you are creating a Pre-BIOS extension you must remeber to handle everything since the BIOS is not installed. Remember there is no stack set up and all jump tables must return values even if zero.

HOO files

Built into the Portfolio's editor is a way of launching external programs. These files are called HOOks and are useful for things like spell checkers, word counters, file information displayers. Basically any program where you can use the ability to interrupt the editor to use as an update on the file you are working on. Unfortunately this only service only extends to the editor.

HOO files are really COM (or RUN) files which have a renamed extender which the editor will search for when the FN6 key is pressed inside the editor. Documentation is contained in the Technical Reference Manual under Interrupt 60 Fn 3h.

New Uploads

Compuserve has held a programming marathon and now yields more than 600 files in the Portfolio roundtable mainly through the efforts of BJ Gleason, Don Messerli, and Dave Stewart. This leads into some uploads of note. Version 4.9 of Pbasic is now available as is Dbshow, and a new extended graphics standard the PGX standard. dbshow allows the PF to use DB3 compatible files, as well as including a the PGC file format to view pictures.

Port-A-Thon

A 24-Hr. Online Marathon called the "Port-a-thon" will be held Friday February 21, 1992, starting at 12:00 noon (PST) on Compuserve.

The Portfolio forum (APORTFOLIO) will be visited continuously by Atari personel, led by Portfolio Marketing Manager Don Thomas. Other luminaries in the Portfolio programming and products world will also be on hand to assist and talk with users of our favorite palmtop. Over 50 prizes will be given away by a variety of companies as well as Atari.

New APB

A new APB (Accessories/Peripherals Bulletin) is now on its way to the printer. If you have information about a Portfolio product which you'd like to have listed in the next APB, be sure to provide it to Don Thomas as soon as possible.

PowerBASIC, Portfolio Chess, Instant Speller, Hyperlist Now Available

The PowerBASIC compiler is now available for the Atari Portfolio. Priced at $99.95, it's a must for Portfolio-Specific development.

Portfolio Chess, offering ten levels of play from Novice to Master, is now available for $49.95.

Instant Speller lets you perform spelling checks on your Portfolio files. Now available at $39.95.

Hyperlist lets you organize any topic in any format you define, and is now available for $49.95.

To order, add $5.00 to the purchase price for shipping and handling. Residents of California, Illinois, and Texas please add 8.25% sales tax.

Send orders to:

Atari Computer
Portfolio Marketing Dept.
1196 Borregas Ave
Sunnyvale, CA 94089-1302

Atari ST/TT Q & A

Mike Fulton

Q: What GEM VDI support is there for the greyscale video mode of the TT030?

A: The Esetgrey() function which puts the TT video into greyscale mode is only supported at the XBIOS and hardware level. VDI functions for setting and requesting color palette settings will not function correctly when the Esetgrey() mode is in effect.

Q: I'm trying to use some of the new FSMGDOS VDI calls, but I'm not getting the results that I expect.

A: One thing that's very important about using the new FSMGDOS functions is making sure that the active font is indeed an FSMGDOS Outline rather than a bitmapped font. The new FSMGDOS calls work only with the outline fonts.

One of the most common errors I've seen so far is using the vst_arbpt() function to set the text size. If the active font is a bitmapped font, then this function will have no effect whatsoever. You may think you've set 72pt text, but the VDI may still be set to 12pt.

Q: How can I detect if FSMGDOS is running, and if a font is an FSMGDOS outline font instead of a bitmapped font?

A: If you place a -2 into register d0 and then do a TRAP #2 call, the return value in d0 will be 0x5F46534D ("_FSM") if FSMGDOS is active. If FontGDOS is active, the return value in d0 will be 0x5F464E54 ("_FNT").

Note that some languages provide a vq_gdos() function which is used to detect GDOS. These functions may not return the correct values for FSMGDOS and FontGDOS. They may detect that some version of GDOS is available, but not which particular one. The values returned by these functions may not be equal to those values shown above.

If you determine that FSMGDOS is not active, you should not make any calls that require FSMGDOS. Note that the bezier curve functions are available with either FSMGDOS or FontGDOS.

Once the program has determined that FSMGDOS is active, then it activates a particular font, it should determine if that font is an FSMGDOS outline font or a bitmapped font.

When you do a vqt_name() call with FSMGDOS active, the intout[32] value will contain a 1 if the font is an FSMGDOS outline font, or a 0 if it is a bitmapped font. Whatever your program uses to set the current font can make this call and save the flag value to indicate to the rest of the program if it should make FSMGDOS calls or not.

Q: I'd like to write a program that plays a different digitized sound everytime something happens an alert box being put up on screen, or a window opening or closing, or a different sound each time a different key is pressed. How do I hook into the system so I can do this and know what's what?

A: First of all, such a program should only be done as a combination of a TSR AUTO-folder program that does the actual work of playing the sounds, and a desk accessory or XCONTROL panel module (or both) which configures the TSR as to which sound to play for which event, and whatever other user-defined settings there might be.

The TSR would grab the Trap #2 vector, which is used to call all GEM VDI and AES functions. The original vector would be saved to be used as the exit point of the TSR's trap handler. (Please note that programs which steal the Trap #2 vector should be placed in the AUTO folder after FSMGDOS.)

The TSR's trap handler would look in register d0 for the AES magic number $C8 (decimal 200). If that value is found, register d1 contains a pointer to the AES parameter block. The trap handler would then obtain the AES function opcode through the following method:

```
move.l d1,a0      ; Get AESPB Ptr
move.l (a0),a0    ; Get CONTROL Ptr
move.w 0(a0),d2   ; Get CONTROL(0)
```

At this point, register d2 would now contain the AES function opcode, and that value would be used to decide what sound to play, if any. After starting the sound, then the trap handler would pass control to the original trap handler so that the AES could execute the requested function.

The above method will not provide any info about keypresses. As of TOS 1.06, there is a new OS vector called kcl_hook at location 0x05B0. If you put the address of your own function into this vector, it will get called each time a key is pressed, and the low byte of register d0 will contain the keyboard scancode (but not the ASCII code) of the key. The remainder of d0 will contain garbage, so be sure to mask it off.

The ASCII value of a key may be different depending on the language version of TOS, so if your function needs to worry about ASCII values, it may wish to use the keyboard lookup table information available through the Keytbl() XBIOS function. (Do this at program initialization, not in the interrupt function.) Note, however, that there is no table for ASCII values generated in conjunction with the Alternate key, and some language versions of TOS use this key to generate certain ASCII values. Your function would have to handle such keystrokes as a special case.

continued on page 8

PowerBASICLibrary for Portfolio Only Programming

copyright © 1992 by BJ Gleason

PowerBASIC, the BASIC compiler for the Portfolio, is a very nice product, limited by support for "Portfolio Only" features. This collection of routines is designed to support built-in features, such as Menus, Screen Saving, Boxes, etc., and also includes support for PGC and PGF graphics.

Here is the complete list of functions provided:

```
box          Display a Box
cmode        Set Cursor Mode
errwin       Display an Error Window
FNcmode%     Get Cursor Mode
FNdsize%     Return Size of RAM Disk (C:)
FNlssize%    Get Logical Screen Size
FNmenu%      Create/Display/Return Menu Item
FNpssize%    Get Physical Screen Size
FNsigned%    Convert a Real to an Integer
FNsmode%     Return Screen Mode
FNspeed%     Get the CPU Speed
FNvspos%     Get Virtual Screen Position
init         Initialize Portfolio Services
meswin       Display a Message Window
noise        Make Noise (Key Click, Beep, Alarm)
off          Turn the Portfolio Off
pgcl         Load a PGC file to Video Memory
pgcs         Save Video Memory as a PGC file
pgfl         Load a PGF file to Video Memory
pgfs         Save Video Memory as a PGF file
rfsh         Refresh the Screen
savres       Save and Restore areas of the screen
smode        Set Screen Mode
speed        Set the CPU Speed
vmove        Move Virtual Screen Position
vspos        Set Virtual Screen Position
```

Since PowerBASIC does not support libraries, the code for these routines will have to be appended onto your source code. Most of these routine are independent of each other, so you need only include those you are using in the program in order to keep the program as small as possible. The only routine that is called by others is the FNsigned% function, which is used to create an integer from a real number. However, if you leave it out when a routines needs it, PowerBASIC will generate an error message.

The only routines that need additional explanation are the PGC/PGF routines. The routines to load them only copy them to video memory, but does not display them. You would have to call RFSH after PGFL or PGCL. The reason for this is to allow one image to "replace" another on the screen. The load and save routines also expect the screen to be in graphics mode (SCREEN 1) before you call.

Here is an example program showing the calling sequence for PGCL:

```
1 input "Enter Filename";f$
if f$="" then end
f$=f$+".PGC"
screen 1
call pgcl(f$)
call rfsh
a$=input$(1)
screen 0
goto 1
end
```

The PowerBASIC compiler was designed to run on the Portfolio, but it will also run on a PC. Since PowerBASIC does not have built in editor, an editor has been developed to allow for quick and easy program development on your desktop.

The PowerBASIC Editor 1.0 (POWED.ZIP) is a full screen text editor, very similar to that of Turbo Pascal 3.0. But it has a couple of built in additions. By pressing ALT-C, you can save your file and invoke the PowerBASIC compiler. While you can compile the file, you can not run it. While this might not appear to be much of a benefit, you do get the advantage of an 80x25 screen, and the editor will automatically point to the line that the compile error appears on.

To use POWED, you need to copy PB.RUN from the PowerBASIC ROM card to your PC and rename it PB.COM. Invoke POWED with the name of the file, and then press ALT-C to compile. When the compile is complete, you will have a .COM file that can be copied to the Portfolio. To speed things up, the <F7> key will invoke FT.COM and send over the file that you just compiled.

Finally, here is the complete list of the library routines. Please feel free to include these in your programs. If you implement any of the other functions, please send them to me, and I will include them in future releases.

All the programs mentioned, and the library routines are available for downloading on Compuserve, in the APORTFOLIO forum.

```
def fnsigned%(unsigned!)
  if unsigned! > 32767 then
     fnsigned% = unsigned! - 65536
  else
     fnsigned% = unsigned!
  end if
end def

sub rfsh shared
'
' copy video memory to LCD screen
'
     reg 1, 4608
     call interrupt 97
end sub

sub pgfl (f$) shared
'
' copy .PGF file to video memory
'
     def seg = 45056
     pg$=""
     open "b",1,f$
     get$ 1, 1920, pg$
     close 1
     for i%= 0 to 1919
          poke i%,asc(mid$(pg$,i%+1,1))
     next i%
end sub

sub pgfs (f$) shared
'
' save video memory to .PGF file
'
     def seg = 45056
     pg$=""
     for i% = 0 to 1919
          pg$ = pg$ + chr$(peek(i%))
     next i%
     open "b",1,f$
     put$ 1, pg$
     close 1
end sub

sub pgcl (f$) shared
'
' copy .PGC file to video memory
'
     def seg = 45056
     offs% = 0
     c$=""
```

```
    pg$ = ""
    open "b",1,f$
    get$ 1, lof(1), pg$
    close 1
    fp% = 4
    do
     cnt% = asc(mid$(pg$,fp%,1))
     fp% = fp% + 1
     if cnt% > 128 then
          cnt% = cnt% - 128
          c% = asc(mid$(pg$,fp%,1))
          fp% = fp% + 1
          for i% = 1 to cnt%
               poke offs%,c%
               offs% = offs% + 1
          next i%
     else
          for i%=1 to cnt%
               c%=asc(mid$(pg$,fp%,1))
               fp% = fp% +1%
               poke offs%,c%
               offs% = offs% + 1
          next i%
     end if
    loop until offs% > 1919
end sub

sub pgcs (f$) shared
'
' copy video memory to .PGC file
'
    def seg = 45056
    offs% = 0
    c$ = ""
    pg$ = "PG" + chr$(1)
    in$ = ""
    run% = 0
    unique% = 0

    do
         do
              run% = 0
              do while ((peek(offs%+run%) = _
              peek(offs%+run%+1)) and (run% < 126) _
              and (offs%+run% < 1919))
                   run% = run% + 1
              loop

              if (run% > 0) then
                   if(unique%<>0) then
                        pg$ = pg$ + chr$(unique%)
                        pg$ = pg$ + in$
                        unique% = 0
                        in$ = ""
                   end if
                   pg$ = pg$ + chr$(run%+129)
                   pg$ = pg$ + chr$(peek(offs%+run%))
                   offs% = offs% + run% + 1
              else
                   in$ = in$ + chr$(peek(offs%))
                   unique% = unique% + 1
                   offs% = offs% + 1
              end if
         loop while((unique% < 127) and (offs% < 1920))
         if(unique%<>0) then
              pg$ = pg$ + chr$(unique% and 127)
              pg$ = pg$ + in$
              in$ = ""
              unique% = 0
         end if
    loop while(offs% < 1920)
    if(unique%) then
         pg$ = pg$ + chr$(unique% and 127)
         pg$ = pg$ + in$
    end if

    open "b",1,f$
    put$ 1, pg$
    close 1
end sub

sub box (x1%, y1%, x2%, y2%, bt%) shared
'
' display a box on the screen.
'    x1, y1 : upper left corner
'    x2, y2 : lower right corner
'    bt-1   : double line box
'    bt-0   : single line box
'
    reg 2,0
    reg 3,x2% * 256 + y2%
    reg 4,x1% * 256 + y1%
    reg 1,2304 + bt%
    call interrupt 96
end sub

sub savres (x1%, y1%, x2%, y2%, ser%, bs%, bo%) shared
'
' screen save/restore
'    x1, y1 : upper left corner, 0 based
'    x2, y2 : lower right corner, 0 based
'    ser  : service
'         0 - Save characters only
'         1 - Save characters and Attributes
'         2 - Restore characters only
'         3 - Restore characters and Attributes
'    bs   : segment to buffer to save/restore screen
'    bo   : offset to buffer
'
'    ss$ = string$(350,0) :' make sure there is enough room
'    bs% = fnsigned%(strseg(ss$))
'    bo% = fnsigned%(strptr(ss$))
'
    reg 2,0
    reg 3,x2% * 256 + y2%
    reg 4,x1% * 256 + y1%
    reg 5,bo%
    reg 8,bs%
    reg 1,2048 + ser%
    call interrupt 96
end sub

sub meswin (x%, y%, ms%, mo%) shared
'
' display a box on the screen.
'    x, y : upper left corner, 0 based
'    ms   : segment to message text
'    mo   : offset  to message text
'    Each line is terminated with a 0
'       text has to be terminated with 00
'    First element is the title
'
'    m$ = "Title" + chr$(0) + "Message" + chr$(0) + chr$(0)
'    ms% = fnsigned%(strseg(m$))
'    mo% = fnsigned%(strptr(m$))
'
    reg 2,0
    reg 4,x% * 256 + y%
    reg 5,mo%
    reg 8,ms%
    reg 1,4608
    call interrupt 96
end sub

sub errwin (x%, y%, ms%, mo%) shared
'
' display a box on the screen.
'    x, y : upper left corner, 0 based
'    ms   : segment to message text
'    mo   : offset  to message text
'       text has to be terminated with 00
'
'    m$ = "Test" + chr$(0) + chr$(0)
'    ms% = fnsigned%(strseg(m$))
'    mo% = fnsigned%(strptr(m$))
'
    reg 2,0
    reg 3,1
    reg 4,x% * 256 + y%
    reg 5,mo%
    reg 8,ms%
    reg 1,5120
    call interrupt 96
end sub

def FNmenu%(x%, y%, top%, las%, bt%, ms%, mo%)
'
' display a box on the screen.
'    x, y : upper left corner, 0 based
'    top  : last top line
'    las  : last selected item
'    bt   : border (0 single, 1 double)
'    ms   : segment to menu text
'    mo   : offset  to menu text
'    Each element is terminated with a 0
'       text has to be terminated with 00
'    First element is the title
'
'    m$="Again?"+chr$(0)+"Yes"+chr$(0)+"No"+chr$(0)+chr$(0)
'    ms% = fnsigned%(strseg(m$))
'    mo% = fnsigned%(strptr(m$))
'
'    returned value
'    -1 - escape pressed
'    int(mc% / 256) = top line
'    mc% mod 256 = selected line
'
    reg 2,0
    reg 4,x% * 256 + y%
    reg 3,top% * 256 + las%
    reg 5,mo%
    reg 6,-1
    reg 8,ms%
```

```
    reg 9,ms%
    reg 1,3840 + bt%
    call interrupt 96
    FNmenu% = reg(1)
end def

sub init shared
'
' Portfolio Service Initialization
' Use once at beginning of program
'
    reg 1, 0
    call interrupt 97
end sub

def FNdsize%
'
' get the size of the RAM Disk
'
    reg 1,2048
    call interrupt 97
    FNdsize% = reg(2)
end def

def FNsmode%
'
' get screen mode
' returns
'      0 : 80 x 25 Static
'      1 : 40 x 8
'      2 : 80 x 25 Tracked
'    128 : Graphics
'
    reg 1,3584
    call interrupt 97
    FNsmode% = reg(4)
end def

sub smode(m%) shared
'
' set screen mode
' m =
'      0 : 80 x 25 Static
'      1 : 40 x 8
'      2 : 80 x 25 Tracked
'    128 : Graphics
'
    reg 1,3585
    reg 4,m%
    call interrupt 97
end sub

def FNcmode%
'
' get cursor mode
' returns
'    0 : cursor off
'    1 : Underline
'    2 : block
'
    reg 1,3840
    call interrupt 97
    FNcmode% = reg(2)
end def

sub cmode(m%) shared
'
' set cursor mode
' m =
'    0 : cursor off
'    1 : Underline
'    2 : block
'
    reg 1,3841
    reg 2,m%
    call interrupt 97
end sub

def FNpssize%
'
' get physical screen size
' rows = int(result / 256)
' cols = result mod 256
'
    reg 1,3328
    call interrupt 97
    FNpssize% = reg(1)
end def

def FNlssize%
'
' get logical screen size
' rows = int(result / 256)
' cols = result mod 256
'
    reg 1,3328
    call interrupt 97
    FNlssize% = reg(4)
end def

def FNvspos%
'
' get virtual screen position
' row = int(result / 256)
' col = result mod 256
'
    reg 1,4096
    call interrupt 97
    FNvspos% = reg(4)
end def

sub vspos(x%, y%) shared
'
' set virtual screen position
' x = row
' y = col
'
    reg 1,4097
    reg 4,x% * 256 + y%
    call interrupt 97
end sub

sub vmove(dir%, dis%) shared
'
' move virtual screen position
' dir = direction
'    1 : up
'    2 : down
'    3 : left
'    4 : right
' dis = distance in characters
'
    reg 1,4352 + dis%
    reg 4,dir%
    call interrupt 97
end sub

sub noise(ty%) shared
'
' make noise
' ty =
'    0 : Key Click
'    1 : Beep
'    2 : Alarm
'
    reg 1,5376 + ty%
    call interrupt 97
end sub

def FNspeed%
'
' get clock tick speed
' returns
'    0 : slow - 1 tick 128 seconds
'    1 : fast - 1 tick per second
'
    reg 1,7680
    call interrupt 97
    FNspeed% = reg(2)
end def

sub speed(m%) shared
'
' set clock tick speed
' m =
'    0 : slow - 1 tick 128 seconds
'    1 : fast - 1 tick per second
'
    reg 1,7681
    reg 2,m%
    call interrupt 97
end sub

sub off shared
'
' turn the portfolio off
'
    reg 1,11520
    call interrupt 97
end sub
```

ABOUT THE AUTHOR

B.J. Gleason is an instructor of Computer Science at The American University in Washington D.C. and he's been programming for over a decade now. He's the author of over four dozen utilities and games, including PBASIC 4.9, the 'freeware' BASIC interpreter designed specifically for the Portfolio. His Email address is BJGLEAS@auvm.american.edu and his Compuserve ID is 75300,2517.

Atari Developer News

ATARI.RSC Staff

METADOS v2.0 Now Available

Metados v2.0 is now available to developers in the download sections of the developer's areas on GEnie and Compuserve.

Metados v2.0 is the software interface to devices such as CDROM drives.

Metados v2.0 includes a driver that can be used with any SCSI CDROM drive that follows the industry standard command set, connected over the TT030's SCSI port. The driver have been tested with Chinon, NEC, and even Apple CDROM players.

A driver that would allow such drives to be connected through a host adapter (such as ICD's) over the ACSI DMA bus is expected to be made available.

VMEbus Information

A number of developers have requested information about the VME expansion bus in the TT030 and Mega STE. For more information, please contact the VMEbus International Trade Association (VITA) at:

VITA
10229 N. Scottsdale Rd., Suite B
Scottsdale, AZ 85253
(602) 951-8866

New Hard Drive Utilities

Version 5.0 of the Atari Hard Drive Utilities and Atari Hard Disk Driver is now available in the download sections of the developer's areas on GEnie and Compuserve.

The new hard disk driver is now configurable to allow it to use non-contiguous device ID numbers.

Atari ST/TT Q & A
Continued from Page 4

Your function can use d0-d2/a0-a2 but should preserve all other registers. The function should either end with an RTS instruction or jump through the previous kcl_hook vector to call the original routine. The routine will be called from the keyboard interrupt, so it should take as little time as possible, and not make any OS calls. The kcl_hook vector is described on page 9 of the STE TOS RELEASE NOTES document.

Atari Computer Corporation
1196 Borregas Ave.
Sunnyvale, CA 94089-1302

ATARI.RSC
The Developer's Resource